

Assignment: Customer Order Manager

**Objective:**

The objective of this assignment is to develop a Customer Order Manager application using Java Fullstack technologies. The application should allow users to create, view, update, and delete customer orders.

**Requirements:**

**Backend Development:**

For Signup API fields should be (name, email, mobile, password – all fields are type string)

For SignIn API username and password to be used.

Accept a JSON payload with the following fields:

"orderNumber" (string): The unique identifier for the customer order (6-digit alphanumeric random ID)

"vendor" (string): The name of the vendor associated with the customer order.

"date" (string): The date when the customer order was created.

"totalAmount" (number): The total amount of the customer order.

“currency” (string) - currency name and symbol

“startDate” (timestamp) - start date of contract

“endDate” (timestampe) - end date of contract

milestone fileds : description(string), date(datetime), amount, status (paid/not-paid- string)

“type” (string) - “Enterprise / Talent”

Implement RESTful APIs to handle the following actions:

Create a new customer order by accepting a JSON payload.

Retrieve a customer order by its order number.

Update an existing customer order by order number.

Delete a customer order by order number.

Use an appropriate database to store customer order information (e.g., MySQL, PostgreSQL, or MongoDB).

Implement validation and error handling in the backend.

**Frontend Development:**

Create a user interface using a front-end framework like Angular, React, or Vue.js.

Implement screens to perform the following actions:

Singup and Singin

View a list of customer orders.

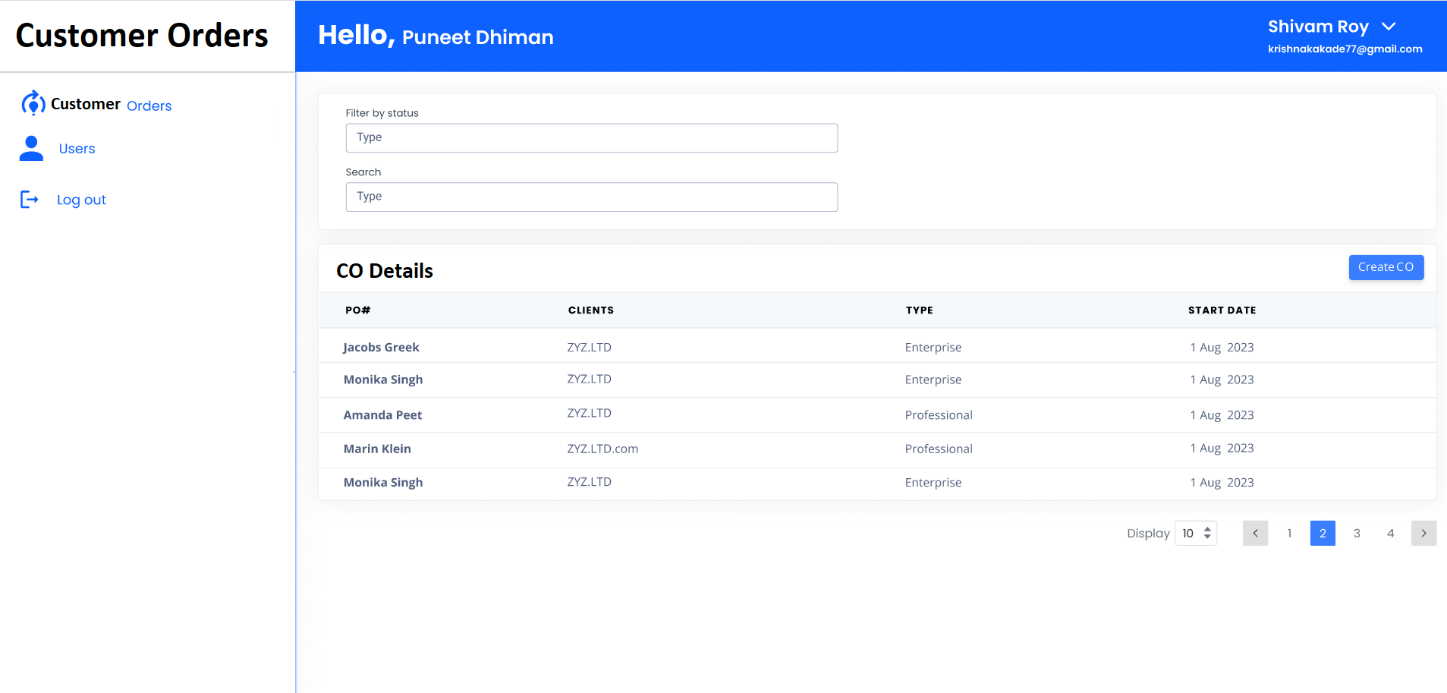
Create a new customer order by providing the required fields.

Update an existing customer order by order number, allowing users to modify the vendor, date, and total amount.

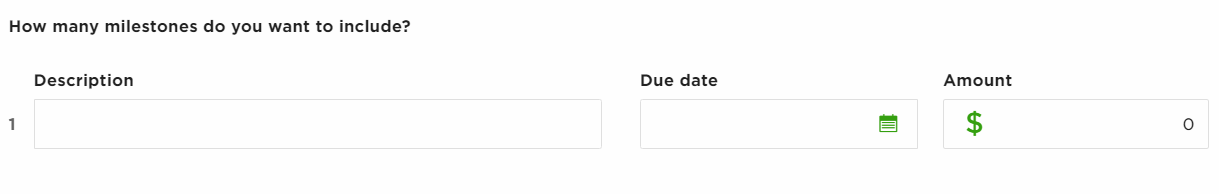
Delete a customer order by order number.

Utilize form validation to ensure data integrity and provide a smooth user experience.

**View Customer Order:**



**Add milestone:**



**View Milestones:**



(do not consider “Review & Pay” and “in escrow” in above image)

**Integration:**

Connect the frontend and backend by making API calls from the frontend to fetch and modify customer order data.

Implement error handling and display appropriate error messages to the user when necessary.

Ensure data consistency between the frontend and backend.

**Additional Features (optional):**

Implement search and filtering options to help users find specific customer orders based on fields like order number, vendor, or date.

Add pagination functionality to display a limited number of customer orders per page.

Include user authentication and authorization to secure the application.

**Documentation:**

Provide clear instructions on how to set up and run the application.

Document the API endpoints and their functionalities.

Explain any additional features or design decisions made during development.

**Submission Guidelines:**Demonstrate the running application.

Share the source code repository (e.g., GitHub) with the instructor.

Include any necessary instructions or documentation for setting up and running the application.